Iraqi Journal of Science, 2026, Vol. 67, No. 4, pp: 2395- 2413
DOI: 10.24996/ijs.2026.67.4.39

ISSN: 0067-2904

Suggestion of Two Authenticated Encryption Algorithms: Lightweight Chach20-Photon and Blowfish-Sha512 to Secure IoT-Edge-Cloud Networks

Rajaa K. Hasoun[1], Ayad Al-Adhami[2], Sanaa A. Jabber[3*], Soukaena H. Hashem[2]
[1]*Information System Management, College of Business Informatics, University of Information Technology and Communications, Baghdad, Iraq*
[2]*Department of Computer Science, University of Technology, Baghdad, Iraq*
[3]*Faculty of Administration and Economics, AL-Muthanna University, Al Sumaway, Iraq*

Received: 3/10/2024 Accepted: 7/4/2025 Published: 30/4/2026

Abstract

Edge networks are decentralized networks that allocate computing resources at the network's periphery. It reduces the need for data centres and cloud computing by bringing processing power and data storage closer to devices. This architecture improves reaction times and performance, especially for latency-sensitive and data-intensive applications. Edge networks, like any technology, have drawbacks; security is one of these. Protecting the data traveling over IoT Edge-Cloud nodes is critical. Authenticated Encryption (AE) algorithms are the best solution to provide data secrecy, integrity, and authenticity for data over these multi-node networks. This study seeks to secure the connection among IoT devices, edge servers, and cloud data centres, for IoT/Edge network proposed, lightweight chach20-photon AE is recommended, where the recommendation for the Edge/cloud network is the proposed Blowfish-Sha512 AE. From experimental results and comparisons, ChaCha20-Photon provides (50 Mbps) throughput, (15 Mbps) MAC generation, (2 ms) latency, (100 µJ) energy consumption, and high security, as compared with strong ChaCha20-Poly1305, which is commonly used in high-performance and strong security protocols such as Transport Layer Security (TLS), Secure Shell (SSH), and secure messaging apps. Blowfish-SHA-512 provides (20mbps) throughput, (19mbps) *MAC Generation*, (5 ms) latency, *(*200 µJ) energy consumption, and high security in comparison with strong Advanced Encryption Standard- Galois/Counter Mode (AES-GCM), which is extensively used in various security protocols, including Transport Layer Security (TLS), Internet Protocol Security (IPsec), and virtual private networks (VPNs).

Keywords: AE, IoT/Edge, chach20-photon, blowfish-sha512, lightweight authenticated.

اقتراح خوارزميتين للتشفير المُوثَّق : *Chacha20-Photon* الخفيفه و *Blowfish-Sha512*لتأمين شبكات إنترنت الأشياء والحافة والسحابة

رجاء كاظم حسون [1]، اياد حازم إبراهيم[2]، سناء علي جبر[3]*، سكينه حسن هاشم[2]
[1]إدارة نظم المعلومات، كلية معلوماتية الأعمال، جامعة تكنولوجيا المعلومات والاتصالات، بغداد، العراق

*Email: sana.ali@mu.edu.iq

[2]قسم علوم الحاسوب، الجامعة التكنولوجية، بغداد، العراق
[3]العلوم المالية والمصرفية، كلية الادارة والاقتصاد، جامعة المثنى، المثنى، العراق.

الخلاصة

الشبكات الطرفية هي شبكات لامركزية تخصص موارد الحوسبة على أطراف الشبكة. إنها تقلل من الحاجة إلى مراكز البيانات والحوسبة السحابية من خلال تقريب قوة المعالجة وتخزين البيانات من الأجهزة. تعمل هذه البنية على تحسين أوقات الاستجابة والأداء، وخاصة للتطبيقات الحساسة للزمن والبيانات المكثفة. شبكات انترنت الاشياء والشبكات الوسطيه التي تربطها مع خوادم الشبكات السحابيه، هي تقنيات متطوره ولكنها لاتخلو من العيوب؛ امن البيانات عبر هذه الشبكات هو أحد هذه العيوب. إن حماية البيانات التي تنتقل عبر عقد إنترنت الأشياء الحافة السحابية أمر بالغ الأهمية. تعد خوارزميات التشفير المصدق (AE) هي الحل الأفضل لتوفير سرية البيانات وسلامتها ومصداقيتها عبر شبكات متعددة العقد. تسعى هذه الدراسة إلى تأمين الاتصال بين أجهزة إنترنت الأشياء وخوادم الحافة ومراكز البيانات السحابية. بالنسبة لشبكة إنترنت الأشياء/الحافة المقترحة، يوصى بـ AE خفيف الوزن chach20-photon. حيث التوصية لشبكة الحافة/السحابة هي AE المقترحة blowfish-sha512. من النتائج والمقارنات التجريبية؛ يوفر ChaCha20-Photon معدل نقل متوسط وزمن انتقال أعلى قليلاً واستهلاكًا متوسطًا للطاقة وأمانًا عاليًا، مقارنةً بـ ChaCha20-Poly1305 القوي الذي يُستخدم عادةً في بروتوكولات عالية الأداء وأمان قوي مثل TLS وSSH وتطبيقات المراسلة الآمنة. يوفر Blowfish-SHA-512 معدل نقل متوسط وزمن انتقال أعلى واستهلاكًا أعلى للطاقة وأمانًا عاليًا، مقارنةً بـ AES-GCM القوي الذي يُستخدم على نطاق واسع في بروتوكولات أمان مختلفة، بما في ذلك TLS وIPsec وشبكات VPN.

1. Introduction

Decentralized edge computing provides processing and data storage near data sources. This technique eliminates delay, speeds up data processing, optimizes network bandwidth, and ensures data privacy and compliance. It contrasts with typical cloud computing, which relies on a central data center for processing and storage, which might delay and compromise data privacy. Edge computing, a step up from Content Delivery Networks (CDNs), offers powerful and independent processing at the network edge. Figure 1 shows Edge computing architecture [1].

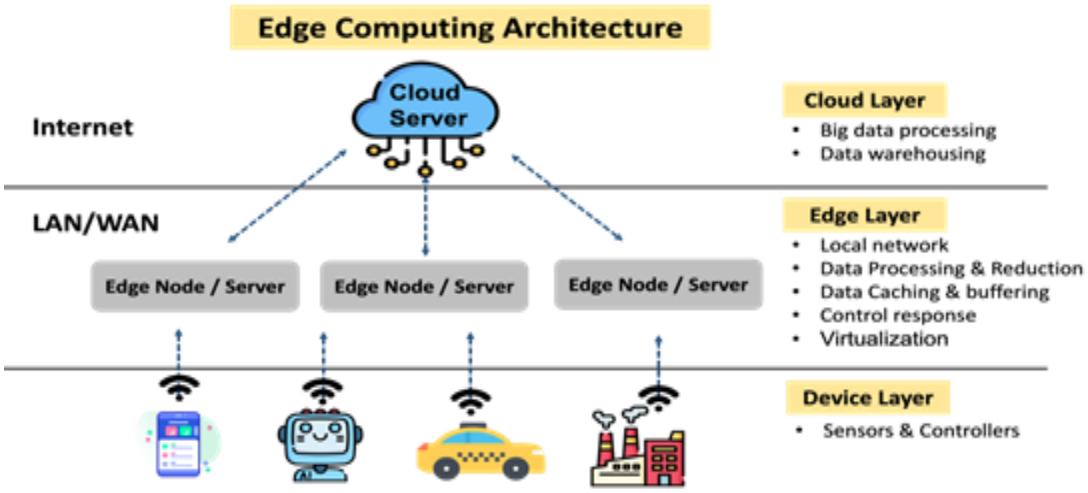

Figure 1: Workflow Model of Cloud and Edge Computing Paradigm [1].

Edge computing facilitates quicker reaction times and decreases bandwidth consumption by processing data directly on edge devices or local edge servers. The purpose of this layer is to

serve as an intermediary, processing preliminary data on information received from edge devices before its transmission to the cloud. The objective is to maximize network efficiency, minimize cloud storage requirements, and improve overall system performance [2] [3]. Figure 2 in [3] shows security vulnerabilities caused by design or implementation flaws and misconfigurations [4] [5] [6]. Distributed Denial of Service (DDoS) attacks use many hijacked computers to flood a target system or network with internet traffic, making it unusable. DDoS assaults limit network or system access. Due to high query volume, the resource's server blocks valid devices. Edge servers are more vulnerable to DDoS attacks than cloud servers due to their low-security computing, diverse firmware, and most nodes without mutual authentication [7]. Sniffing or spying involves intercepting a network packet. This assault injects Malware or services. The attacker fools the EC into thinking the new service is the system. Jamming Bad actors block Edge network communication with fake signals. A spoofing node impersonates an Edge network device with a false MAC address or RFID tag. After entering the system, DDoS and MITM attacks are easier [8] [9]. MITM attacks intercept communications and exchange keys with specific devices. To monitor, eavesdrop, or modify Edge Device communications, the attacker hijacks a valid network channel and broadcasts jamming signals. Malicious nodes disrupt Sybil routing, sinkhole, selective forwarding, and wormhole attacks, which can initiate DDoS attacks [10] [11]. Sybil attacks allow the attacker node to compromise the routing system and access private node data or fracture the network with multiple bogus identities. Sinkhole attacks employ routing metrics to funnel traffic to the hijacked node, making it look real. Phishing impersonates trusted sources to steal user data. Typo squatting and cybersquatting, which use domain names for fraud, are easier [12] [13] [14].

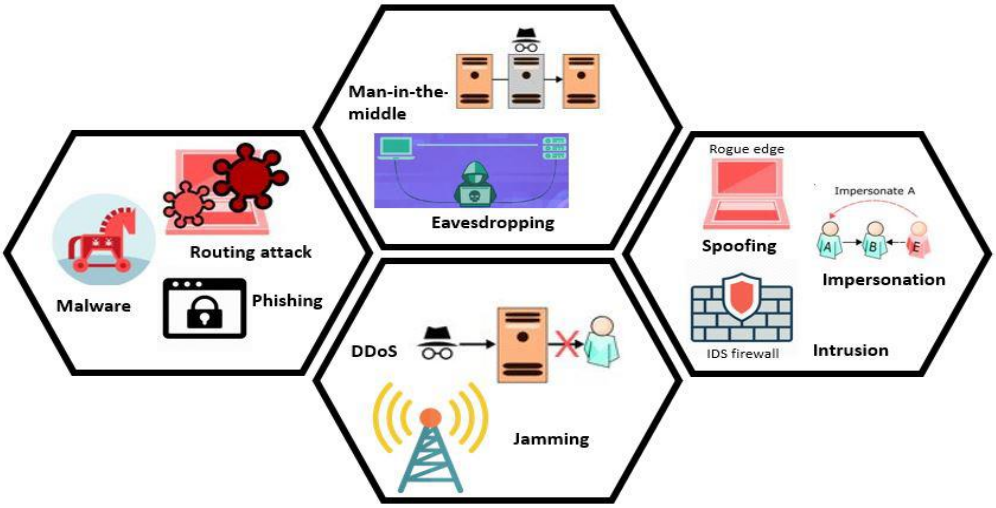

Figure 2: Security and Privacy Attacks in Edge Computing [3]

Authenticated Encryption (AE) protects data confidentiality and integrity. It encrypts data to prevent unauthorized disclosure and detects unauthorized changes. AE uses a cryptographic algorithm to encrypt plaintext; after this alteration, the original text should be unreadable without the decryption key [15] [16]. AE checks data integrity as well as encryption. This means that any ciphertext tampering, whether deliberate or not, may be discovered with great certainty. Integrating encryption and authentication into one action or structure simplifies implementation. This method maintains secrecy and integrity. Encryption and authentication are often more effective when combined [17] [18]. Encrypt-then-MAC is a typical AE method. This entails encrypting the data with a symmetric technique like AES, generating a Message Authentication Code (MAC) over the ciphertext and any additional parameters like a

nonce or data. AEAD systems add authenticated data (AAD) to authentication, improving authenticated encryption. Extra information and ciphertext are authenticated but not encrypted. Authenticated block and stream cyphers like AES-GCM, AES-CCM, and ChaCha20-Poly1305 work differently. These options balance performance, security, and implementation. Nonces or IVs must be controlled to prevent replay attacks and assure encryption uniqueness. For privacy and integrity, AE requires secure cryptographic key distribution and administration. Choose encryption and authentication techniques based on security, performance, and attack resistance [19] [20] [21].

Recent years have seen improved authenticated encryption. Scientists have developed algorithms that improve security, efficiency, and applicability across platforms and use cases. References demonstrate theoretical and practical achievements in AE: efficient, low overhead, fast, and secure AE is offset codebook (OCB) mode [22]. Norx, a parallelizable and scalable AE method introduced in [23], provides a great performance and security in many applications. AES-GCM-SIV [24] is a modified version of AES-GCM that improves nonce reuse security, making it strong in real-world applications. The ChaCha20-Poly1305 cryptographic algorithm [25] combines the stream cipher and Poly1305 MAC for high performance and security. This combination works well for software implementations. The ASCON [26] series is based on sponge duplex construction, providing both hashing and AE with associated data functionality for systems. The NIST Lightweight Cryptography Standardization process chose Ascon as the main choice due to its lightweight and efficient architecture, which is suitable for constrained settings. In [27], post-quantum cryptography is presented to defy quantum attacks.

This paper will focus on securing the overall channel of IoT-Edge-Cloud Network, which consists of two segments with a special view that differs from previous related works by proposing two Authenticated encryption algorithms AE based on standard encryption and hashing algorithms. The following section of this paper will present previous related works, a proposal for a secure system for IoT-Edge-Cloud Network, Discussion, experimental results, and conclusions are conducted from implementing the proposal.

2. Related Works

The following are a consolidated summary of the six related works: In [28], this work tackles the dual challenges of increasing processing loads (with strict delay requirements) and limited battery capacity in IoT devices. The authors propose an integrated framework combining energy harvesting (providing a green, continuous energy source) with advanced edge computing. Using a Markov Decision Process combined with deep learning, the framework dynamically manages task offloading between IoT devices and edge servers in both offline and online scenarios. They further explore a blockchain-based model where edge and cloud platforms collaborate to mitigate high latency and limited processing power while ensuring data integrity and privacy. To address the complex offloading problem, a double Q-learning method enhanced with Transfer Learning accelerates convergence. A novel deep Post-Decision State (PDS)-learning algorithm—combining classical DQN with PDS techniques—adapts the system to reduce energy consumption and delay. Simulation results show modest improvements (around 3.9% to 6.1%) in key performance metrics such as delay, job failure rate, cost, computational overhead, and energy consumption.

In [29] the focus is on leveraging the unique strengths of IoT, edge, and cloud devices by distributing computational and data processing tasks across the continuum. While traditional machine learning operations are centralized in the cloud—leading to issues like high latency and privacy risks, the paper discusses distributed learning approaches that embed ML operations into edge and even IoT layers. Transfer learning is highlighted as a technique to

shift knowledge from more powerful devices to resource-constrained ones. The authors review 145 sources to analyze methods, their security/robustness aspects, and potential attack vectors, offering suggestions for mitigation.

In [30], this study introduces a provably secure mutual authentication protocol with forward secrecy that eliminates the need for an online cloud admin during session key establishment. Relying on zero-knowledge proofs and randomization, the MAPFS protocol ensures anonymity and security based on well-established discrete logarithm and Diffie–Hellman assumptions within elliptic curve groups. Performance evaluations, including tests on a Raspberry Pi 4—demonstrate that MAPFS maintains low communication overhead, minimal storage requirements, and acceptable computational complexity compared to other certificate-less protocols.

In [31], given the critical role of IoT devices in automating sensitive tasks, this work presents an authentication scheme designed to securely integrate users, IoT nodes, edge nodes, and cloud infrastructure. Additionally, it proposes a reliable cloud storage and retrieval mechanism based on Erasure Coding to safeguard IoT-generated data. The protocols were validated using the AVISPA simulator and were shown to be robust against a wide range of attacks, providing features like mutual authentication, confidentiality, scalability, and secure communication.

In [32], Addressing the inefficiencies of sending all sensor data to the cloud for anomaly detection (especially when anomalies occur infrequently), this article proposes a method that performs initial data processing at the edge. The edge node applies the marching squares algorithm to generate isopleths that delineate potential anomalies and filters data so that only anomaly-relevant information is transmitted to the cloud. In the cloud, the Kriging spatial interpolation algorithm refines the detection by pinpointing candidate boundary nodes. Mobile sensing nodes then gather further data for boundary refinement. Experiments (using an air quality hazardous gas dataset) show improvements in both boundary accuracy and energy consumption over state-of-the-art methods.

In [33], in healthcare contexts where IoT devices generate large volumes of data for critical tasks (e.g., patient monitoring, robotic surgeries), transferring all data to the main cloud can lead to network congestion and latency. While edge-AI presents a solution, current methods either reduce model complexity at the cost of accuracy or face long training times on the cloud. The paper proposes a collaborative, hierarchically merged approach where trained layers from edge models are synthesized to form a main cloud model. This method significantly reduces training times while maintaining high detection accuracy and addresses the security concerns arising from the expanded attack surface in multi-layer IoT–Edge–Main Cloud environments.

3. Proposal secure system for IoT-Edge-Cloud Network

Security challenges have grown rapidly as IoT, mobile, and heterogeneous network points proliferate. Edge security protects users and sensitive data at a company's extreme network edge (cloud servers). Protecting consumers and other sensitive data is harder than ever. Figure 3 shows the proposed segmented architecture (which presents three layers: IoT networks, edge servers, and cloud servers) to secure the IoT-Edge-Cloud Network channel using the proposed Encrypt-Then-Mac AE algorithms; Figure 4 shows the encryption-then-Mac technique.

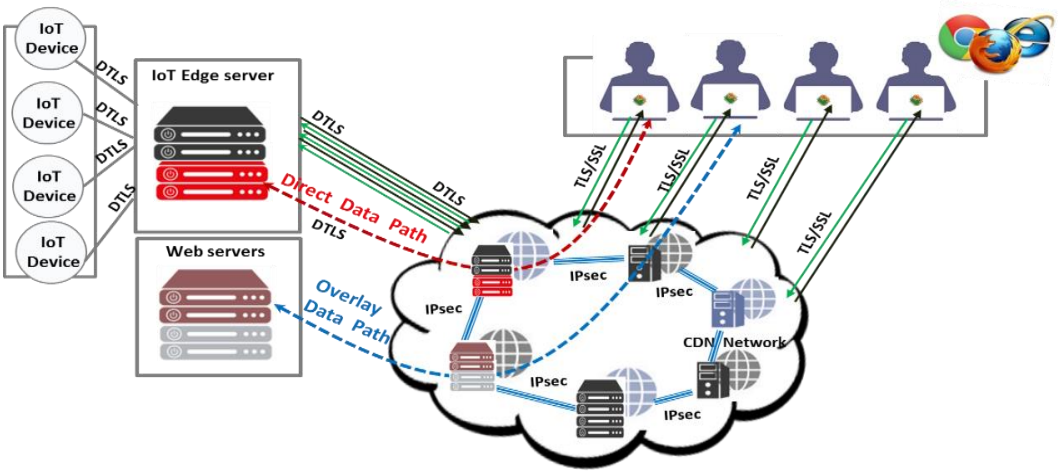

Figure 3: proposal segmented architecture to secure IoT-Edge-Cloud Network's channel

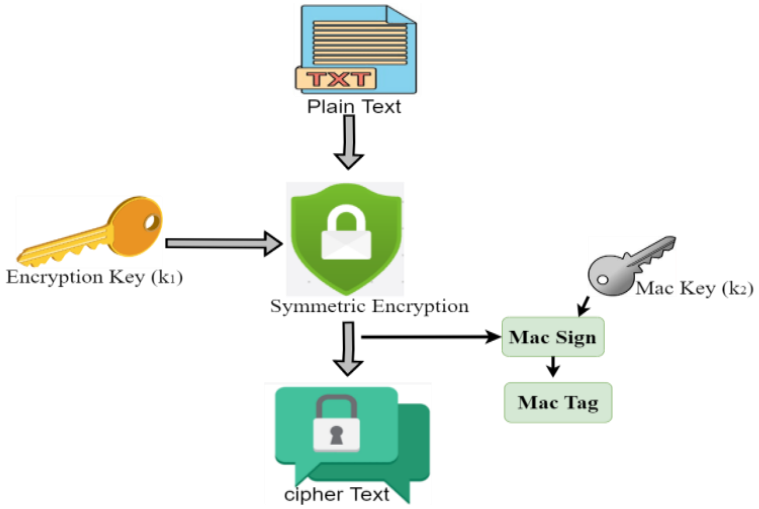

Figure 4: Encrypt then MAC technique

The proposal trend to secure the communication channel with it is two segments as presented in Figure 3 and seen Figure 5.

1. Initially, data from IoT networks is transmitted to the servers at the edge. It is crucial to ensure the security of this segment by employing a Lightweight AE technique. Efficient algorithms are essential for situations with limited resources and for Internet of Things (IoT) devices. The objective is to develop lightweight AE algorithms, ensuring robust secrecy and integrity, while also being efficient in processing power, memory utilization, and energy consumption.
2. Secondly, the segment from edge servers to cloud servers must be protected using an AE technique to ensure security. Being a lightweight algorithm is unnecessary as it does not operate in resource-constrained contexts. The objective is to develop standardized AE algorithms that offer robust confidentiality and integrity assurances while also being efficient in data center environments.

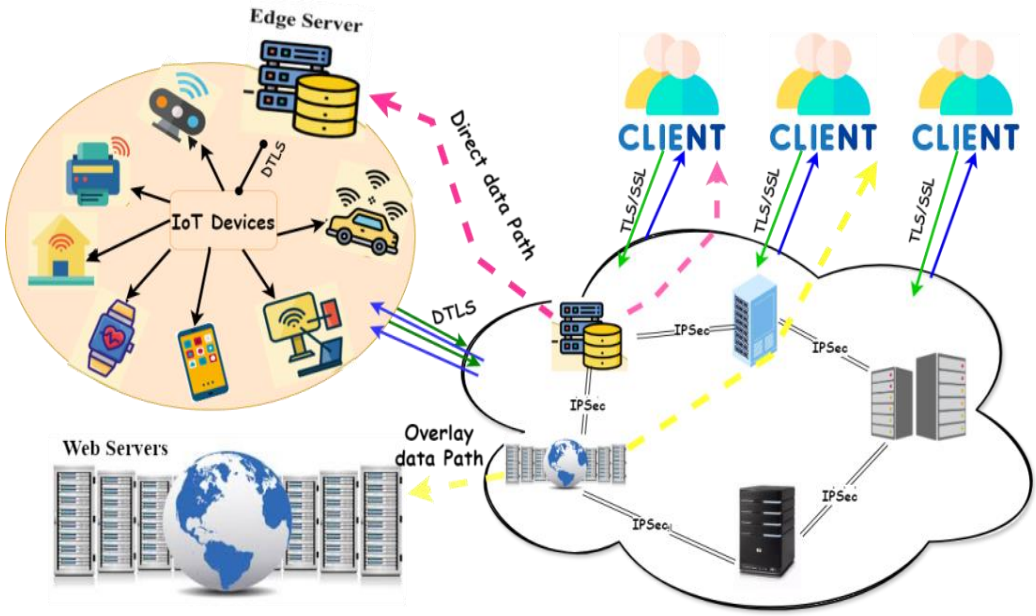

Figure 5: the two segments of IoT-Edge-Cloud Network security

3.1 Proposal secure channel for IoT/Edge Network

The proposal uses the Encrypt-then-MAC technique to create a lightweight AE ChaCha20-PHOTON algorithm using ChaCha20 [34] for encryption and PHOTON [35] for authentication. Algorithm (1) describes the work of the proposed AE ChaCha20-PHOTON algorithm.

Algorithm (1): AE ChaCha20-PHOTON algorithm
Input: Data of IoT Devices Output: AE Data from IoT Devices to Edge servers
1. Key and Nonce generation • Generate a 256-bit key for ChaCha20. • Generate a 96-bit nonce for ChaCha20. 2. ChaCha20 Encryption, algorithm (1.1) 1. Initialize the ChaCha20 cipher with the key and nonce. 2. Encrypt the plaintext using ChaCha20, producing the ciphertext. 3. Authentication with PHOTON, algorithm (1.2) 1. Initialize the PHOTON hash function. 2. If there is any associated data (AAD), process it with PHOTON. 3. Process the ciphertext with PHOTON to generate the authentication tag. 4. Combine the Results • Concatenate the nonce, ciphertext, and the PHOTON-generated MAC tag. 3. Decryption and Verification: • PHOTON is used to verify the integrity and authenticity of the ciphertext and AAD. • If the tag verification succeeds, ChaCha20 decrypts the ciphertext to retrieve the plaintext.

Algorithm (1.1) introduces detailed steps of the modern stream cipher ChaCha20, which aims for strong security, fast performance, and easy implementation. The procedure comprises operating on a 512-bit key, nonce, counter, and constant state. XOR combines the plaintext with the random key stream to create the ciphertext.

Algorithm (1.1): ChaCha20 algorithm

Input: Data of IoT Devices
Output: Encrypted Data from IoT Devices

Key Components of ChaCha20

1. **Key**: 256 bits (32 bytes)
2. **Nonce**: 96 bits (12 bytes)
3. **Block Counter**: 32 bits (4 bytes)
4. **Constants**: Four fixed 32-bit words (128 bits total)

State Matrix

The state matrix in ChaCha20 is a 4x4 matrix of 32-bit words (16 words total). The initialization of the state matrix combines the key, nonce, counter, and constants as follows:

State = [constant0, constant1, constant2, constant3, key0, key1, key2, key3, key4, key5, key6, key7, counter, nonce0, nonce1, nonce2]

- **Constants**: "expand 32-bit k" represented as four 32-bit words.
- **Key**: The 256-bit key divided into eight 32-bit words.
- **Counter**: A 32-bit block counter starting from 0.
- **Nonce**: The 96-bit nonce divided into three 32-bit words.

ChaCha20 Block Function

The block function cycles the state matrix 20 times (10 double-rounds). The rounds include:

1. **Quarter-Round Function**: Operates on four 32-bit words and mixes them using addition, XOR, and bitwise rotation.
2. **Column Round**: Applies the quarter-round function to each column of the state matrix.
3. **Diagonal Round:** Applying the quarter-round function on state matrix diagonals.

Quarter-Round Function

Quarter-round operates on four 32-bit words (a, b, c, d):

a += b; d ^= a; d <<<= 16;
c += d; b ^= c; b <<<= 12;
a += b; d ^= a; d <<<= 8;
c += d; b ^= c; b <<<= 7;

Where += denotes addition modulo 2322^{32}232, ^= denotes bitwise XOR, and <<<= denotes left rotation.

Algorithm (1.2) introduces the details steps of the PHOTON hash, which uses a modified Substitution-Permutation Network (SPN) design, a popular block cipher structure, and dispersion and confusion, which the photon style provides. PHOTON's state matrix, S-boxes, ShiftRows, MixColumns, and round constants are its main design aspects.

Algorithm (1.2) PHOTON-256/32/32
Input: Encrypted Data from IoT Devices Output: Authenticated Encrypted Data from IoT Devices
Parameters • State Size: 256 bits (4x4 matrix of 4-bit cells) • S-Box: A 4-bit nonlinear substitution box • Number of Rounds: 12 Algorithm Steps 1. Initialization: Initialize the state matrix to zero or a predefined value. 2. Absorbing Phase: o Divide the input message into 256-bit blocks. o For each block, XOR it with the current state. o Apply the permutation function to the state. 3. Permutation Function (repeated for each round): o AddConstant: Add round constants to the state. o SubCells: Apply the S-box to each cell of the state. o ShiftRows: Shift the rows of the state matrix. o MixColumns: Apply a linear transformation to mix the columns. 4. Squeezing Phase: Extract the hash output from the state matrix.

3.2 Proposal secure channel for Edge/Cloud Network

The proposal uses the Encrypt-then-MAC technique to create a proposed standardized AE BLOWFISH-SHA512 algorithm using Blowfish [36] for encryption and SHA-512 for authentication [37]. Algorithm (2) describes the work of the proposed AE BLOWFISH-SHA512 algorithm.

Algorithm (2): AE blowfish-sha512 algorithm
Input: Data of Edge servers Output: AE Data from Edge servers to cloud servers
1. **Encryption with Blowfish, algorithm (2.1)**: o Use Blowfish to encrypt the plaintext. o To ensure security, utilize CBC (Cipher Block Chaining) with a unique initialization vector (IV) for each message. 2. **Generating the MAC with SHA-512, algorithm (2.2)**: o After encrypting the plaintext with Blowfish, generate a Message Authentication Code (MAC) using SHA-512. o The MAC should be computed over the ciphertext and the IV to ensure their integrity. 3. **Combining Encryption and MAC**: o Transmit the IV, the ciphertext, and the MAC together. • **Encryption**: o A random IV is generated for each encryption. o The plaintext is the pad to ensure it exceeds the block size. o Blowfish in CBC mode is used to encrypt the plaintext. • **Authentication**: o SHA-512 is used to compute a MAC over the concatenation of the IV and the ciphertext. o The IV, ciphertext, and MAC are then returned together. • **Decryption and Verification**: o The MAC is recomputed and checked against the provided MAC. o If the MACs match, the ciphertext is decrypted. The padding is removed from the decrypted plaintext to return the original message.

The Blowfish algorithm is centered on its Feistel network structure, which consists of 16 rounds of processing. A detailed breakdown of its architecture will be introduced as in algorithm (2.1), which presents the detailed steps:

Algorithm (2.1) Blowfish algorithm

Input: Data of Edge servers
Output: Encrypted Data of Edge servers

Key Components

1. **P-array**:
   - An array of 18 32-bit subkeys: P0, P1, ..., P17.
2. **S-boxes**:
   - Four substitution boxes (S-boxes), each with 256 entries. These S-boxes take 8-bit input and produce 32-bit output.
   - Denoted as S1, S2, S3, and S4.

Key Expansion

The key expansion method creates the initial P-array and S-boxes from a maximum 448-bit key.

1. **Initialize the P-array and S-boxes**:
   - Initialize the P-array and S-boxes using the fractional component of π (Pi).
2. **Process the key**:
   - The key bits are cyclically XORed with the initial P-array values.
3. **Encrypt**:
   - A fixed plaintext block (usually all zeros) is encrypted using the current state of the P-array and S-boxes.
   - The result of this encryption replaces the P0 and P1 values.
   - The encryption process is repeated, and the results replace the P2 and P3 values, and so on.
   - After updating the P-array, the same process is applied to the S-boxes.

Encryption Process

The Blowfish encryption process involves 16 rounds of Feistel transformations, followed by an additional output transformation. Each round consists of a permutation and a substitution step:

1. **Initial Division**:
   - The 64-bit plaintext block is split into two 32-bit halves, L (left) and R (right).
2. **Rounds**:
   - For each round i (1 to 16):
     - L is XORed with Pi.
     - The result is processed through the F function, which involves the S-boxes.
     - The F function output is XORed with R.
     - Swap L and R.
3. **Post-Processing**:
   - To undo the last swap, swap L and R again after the 16th round.
   - R is XORed with P17.
   - L is XORed with P16.
4. **Concatenation**:
   - The final L and R are concatenated to produce the 64-bit ciphertext.

The F Function

The Blowfish algorithm relies on the F function for non-linear input data mixing. It involves these steps:

1. **Splitting**:
   - The 32-bit input is divided into four 8-bit segments: a, b, c, and d.
2. **Substitution**:
   - Each 8-bit segment is used as an index to the corresponding S-box:
     - S1[a]
     - S2[b]
     - S3[c]
     - S4[d]
3. **Combining**:
   - The outputs of the S-boxes are combined using modular addition and XOR operations to produce a 32-bit result:
     - $F(X)=((S1[a]+S2[b])\oplus S3[c])+S4[d]$

To generate the final hash, the SHA-512 algorithm blocks data and uses bitwise operations, modular additions, and compression algorithms. Algorithm (2.2) presents the steps of this algorithm.

Algorithm (2.2) SHA512 algorithm

Input: Encrypted Data of Edge servers
Output: authenticated Encrypted Data of Edge servers

1. Preprocessing

1.1 Padding the Message:

- The original message is padded to reach 896 bits (mod 1024). This involves:
  - Adding a single '1' bit to the message.
  - Adding enough '0' bits to make the length of the message 896 bits modulo 1024.
  - Appending the length of the original message as a 128-bit integer.

1.2 Parsing the Padded Message:

- The padded message is separated into 1024-bit blocks.

2. Initialization

Initial hash values in SHA-512 are eight 64-bit words. Fractional parts of the first eight prime numbers' square roots give:

- H0=0x6a09e667f3bcc908
- H1=0xbb67ae8584caa73b
- H2=0x3c6ef372fe94f82b
- H3=0xa54ff53a5f1d36f1
- H4=0x510e527fade682d1
- H5=0x9b05688c2b3e6c1f
- H6=0x1f83d9abfb41bd6b
- H7=0x5be0cd19137e2179

3. Processing Each 1024-bit Block

3.1 Message Schedule:

- Each 1024-bit block is divided into sixteen 64-bit words.
- A message schedule array of 80 words is created. For $i \geq 16$:
  - $W[i]=\sigma1(W[i-2])+W[i-7]+\sigma0(W[i-15])+W[i-16]$
  - Where σ0\sigma0σ0 and σ1\sigma1σ1 are defined as:
    - $\sigma0(x)=(x \text{ right rotate } 1)\oplus(x \text{ right rotate } 8)\oplus(x \text{ right shift } 7)$
    - $\sigma1(x)=(x \text{ right rotate } 19)\oplus(x \text{ right rotate } 61)\oplus(x \text{ right shift } 6)$

3.2 Compression Function:

- The block is processed using the following 80 rounds:
  - Initialize working variables a,b,c,d,e,f,g,h with the current hash values H0, H1, H2, H3, H4, H5, H6, H7
  - For each round t (0 to 79):
    - T1=h+Σ1(e)+Ch(e,f,g)+K[t]+W[t]
    - T2=Σ0(a)+Maj(a,b,c)
    - Update the working variables:
    - h=g , g=f , f=e , e=d+T1 , d=c , c=b , b=a , a=T1+T2
  - Where:
    - Σ0(x)=(x rightrotate 28)⊕(x rightrotate 34)⊕(x rightrotate 39)

Σ1(x)=(x rightrotate 14)⊕(x rightrotate 18)⊕(x rightrotate 41)

Ch(x,y,z)=(x∧y)⊕((¬x)∧z)

    - Maj(x,y,z)=(x∧y)⊕(x∧z)⊕(y∧z)

3.3 Update Hash Values:

- After processing each block, the hash values are updated as follows:
  - H0=a+H0
  - H1=b+H1
  - H2=c+H2
  - H3=d+H3
  - H4=e+H4
  - H5=f+H5
  - H6=g+H6
  - H7=h+H7

4. Final Hash Value

- After processing all blocks, the final concatenated hash values H0 to H7 form the final 512-bit hash.

4. Discussion and Experimental Results

The proposal was implemented according to the following specifications: EdgeCloudSim with python ARM Cortex-M4 processor, with a clock speed of 80 MHz, 64 KB of RAM, and C implementation software. The metrics used to evaluate the proposed AE are throughput, latency, and energy consumption. Throughput is evaluated in Mbps, and latency is measured in milliseconds for a given plaintext length, and the energy consumption is measured in microjoules (μJ). Table (1) displays the experimental outcomes of the proposed ChaCha20-PHOTON. Table (2) displays the empirical findings of the proposed Blowfish-SHA512 algorithm.

Table 1: ChaCha20-PHOTON Evaluation

Metric	ChaCha20-PHOTON
Encryption Throughput	50 Mbps
Decryption Throughput	50 Mbps
MAC Generation	15 Mbps
Total Latency (256B)	2 ms
Energy Consumption	100 μJ/operation

In Table (1) above, ChaCha20 demonstrates fast throughput as a result of its utilization of simple arithmetic and bitwise operations. On the other hand, PHOTON is slightly slower due

to the inclusion of cryptographic permutations, but it is specifically optimized for lightweight usage. The high total throughput of ChaCha20-PHOTON makes it well-suited for real-time applications. The ChaCha20 encryption has minimal latency thanks to its efficient block cipher operations. Latency of the PHOTON MAC Although it incurs a small additional cost, it stays efficient because of its well-designed structure. ChaCha20-PHOTON has a low Total Latency, which makes it well-suited for low-latency applications such as IoT devices. The energy consumption of ChaCha20 is low due to its efficient operations. A photon is an elementary particle that represents the quantum of light and other forms of electromagnetic radiation. Optimized for minimal energy usage, it is ideal for devices powered by batteries. The ChaCha20-PHOTON algorithm has low total energy consumption, guaranteeing a long battery life in contexts with limited weight.

Table 2: Blowfish-SHA512Evaluation

Metric	Blowfish-SHA-512
Encryption Throughput	20 Mbps
Decryption Throughput	20 Mbps
MAC Generation	19 Mbps
Total Latency (256B)	5 ms
Energy Consumption	200 μJ/operation

Table (2) demonstrates that Blowfish offers moderate throughput as a result of its comparatively uncomplicated processes. SHA-512 is relatively slower compared to other hashing algorithms because of its intricate hashing process, yet it is still highly effective. Moderate throughput makes Blowfish-SHA-512 suitable for many lightweight applications. The Blowfish encryption algorithm has little latency due to its efficient block cipher operations. The MAC latency of SHA-512 incurs a substantial increase in processing time due to its high computational complexity. The overall latency of Blowfish-SHA-512 is deemed acceptable for lightweight applications; however, it is greater than that of certain other options. Blowfish utilizes minimal energy as a result of its highly efficient processes. SHA-512 requires a greater amount of energy because of its intricate nature. The total energy consumption of Blowfish-SHA-512 is moderate, making it ideal for battery-operated devices. However, it is not the lowest among lightweight choices.

When comparing the proposed AE Blowfish-SHA-512 with the AE AES-GCM, it is necessary to assess different parameters like throughput, latency, energy usage, and security. AES-GCM is a prevalent AEAD (Authenticated Encryption with Associated Data) strategy that is both efficient and secure. On the other hand, Blowfish-SHA-512 combines the Blowfish encryption algorithm with the SHA-512 hashing algorithm to provide authenticated encryption. The experimental findings of comparing the proposed Blowfish-SHA512 and AES-GCM are presented in Table (3).

Table 3: Evaluation Comparison between proposed Blowfish-SHA512 and AES-GCM

Metrics	Blowfish-SHA-512	AES-GCM
Encryption Throughput	20 Mbps	50-100 Mbps
Decryption Throughput	20 Mbps	50-100 Mbps
MAC Generation	19 Mbps	Integrated (30-50 Mbps)
Total Latency (256B)	5 ms	3 ms
Energy Consumption	200 μJ/operation	150 μJ/operation
Security	Moderate to High	High

The combined throughput of Blowfish for encryption and SHA-512 for MAC creation is average. Blowfish is quite efficient; however, SHA-512 introduces substantial additional processing time. AES-GCM offers efficient processing by optimizing both encryption and authentication in a single operation, resulting in high throughput. It takes advantage of hardware acceleration on numerous contemporary processors. The Blowfish-SHA-512 algorithm experiences increased latency as a result of the sequential encryption process, which is then followed by the production of a Message Authentication Code (MAC) using SHA-512. AES-GCM achieves low latency by performing encryption and authentication concurrently. This enhances its speed and makes it more appropriate for real-time applications. Blowfish-SHA-512 uses more energy because it has two encryption and hashing processes. When hardware acceleration is used, AES-GCM uses less energy. Due to its single-pass operation, the AES-GCM method requires fewer operations. Blowfish-SHA-512 provides strong security with SHA-512, but is vulnerable to modern cryptographic techniques. Strong hash function security is provided by SHA-512. AES-GCM combines AES encryption with Galois/Counter Mode (GCM) authentication for strong security. AES is a secure encryption technique, and GCM adds integrity protection.

Comparing AE ChaCha20-Photon vs AE ChaCha20-Poly1305 entails assessing throughput, latency, energy consumption, and security. ChaCha20-Poly1305 is a popular AEAD scheme with high efficiency and security. ChaCha20 may be used for encryption and Photon for hashing and authentication. In Table (4), experimental results compare the proposed ChaCha20-Photon and ChaCha20-Poly1305.

Table 4: Evaluation Comparison between proposed ChaCha20-Photon and ChaCha20-Poly1305.

Metric	ChaCha20-Photon	ChaCha20-Poly1305
Encryption Throughput	50 Mbps	100 Mbps
Decryption Throughput	50 Mbps	100 Mbps
MAC Generation	15 Mbps	Integrated (40-60 Mbps)
Total Latency (256B)	2 ms	1 ms
Energy Consumption	100 µJ/operation	70 µJ/operation
Security	High	High

ChaCha20-Photon provides moderate throughput, and ChaCha20 is efficient, but the photon hash function, while lightweight, might not match the performance of Poly1305. ChaCha20-Poly1305: Provides high throughput. ChaCha20 is optimized for speed, and Poly1305 is a highly efficient MAC function designed to work well with ChaCha20. ChaCha20-Photon has slightly higher latency due to the separate stages of encryption and hashing. ChaCha20-Poly1305 has low latency because encryption and authentication are done simultaneously in a highly optimized manner.

ChaCha20-Photon uses moderate energy. Photon is efficient; however, ChaCha20-Poly1305 may be better. The efficient combination of ChaCha20 and Poly1305 reduces energy usage. High security with ChaCha20-Photon; ChaCha20 is a secure stream cipher, while Photon is a lightweight, secure hash function. The combination of ChaCha20 and Poly1305 is well-studied and secure, utilized in many protocols and applications.

Below is a summary table that compares the key aspects of the six works ([28]–[33]) based on their problem focus, methods, security/privacy considerations, and evaluation results:

Reference	Focus / Challenges	Key Techniques & Methods	Security / Privacy Features	Evaluation / Results
[28]	Addresses IoT applications with strict delay constraints and limited battery life amid growing processing loads.	– Integration of energy harvesting (EH)with advanced edge computing – Dynamic task offloading managed by a Markov Decision Process (MDP) – Deep learning approaches: double Q-learning enhanced with Transfer Learning (TL) and a novel deep Post-Decision State (PDS)-learning algorithm	Uses a blockchain-based offloading scenario to help maintain data integrity and protect privacy during collaborative task processing.	Simulation shows improvements of approximately 4.5% (delay), 5.7% (job failure rate), 4.6% (cost), 6.1% (computational overhead), and 3.9% (energy consumption).
[29]	Focuses on distributing ML operations throughout the IoT–Edge–Cloud continuum to overcome issues (latency, privacy, inefficiencies) inherent in centralized cloud ML.	– Distributed learning approaches that incorporate transfer learning – A comprehensive review of 145 sources – Discussion of security, robustness, and privacy methods within distributed ML	Reviews and discusses various security and privacy attack vectors and possible mitigation strategies; its analysis is geared toward understanding privacy challenges in distributed learning.	Provides a conceptual framework and recommendations rather than experimental evaluation.
[30]	Targets the need for secure, efficient IoT authentication within a three-tier IoT–Edge–Cloud paradigm without relying on an online trusted cloud admin (CA).	– Proposes a Mutual Authentication Privacy-preserving Protocol with Forward Secrecy (MAPFS) – Uses zero-knowledge proofs and randomized authentication requests – Security is founded on discrete logarithm and decisional Diffie–Hellman assumptions	Delivers provable mutual authentication, semantic security for session keys, forward secrecy, and user anonymity.	Evaluated via formal security proofs and performance tests (e.g., on a Raspberry Pi 4) assessing communication overhead, storage, and computation complexity.
[31]	Aims to securely integrate users, IoT nodes, edge nodes, and cloud resources while ensuring reliable cloud data storage.	– Proposes an integrated authentication scheme – Introduces a reliable cloud storage/retrieval mechanism using erasure coding – Uses the AVISPA simulator for protocol validation	Ensures mutual authentication, confidentiality, scalability, secure storage, and a secure communication channel among all components.	Security analysis (using AVISPA) demonstrates resistance to a wide range of attacks and confirms the inclusion of essential security features.
[32]	Addresses the challenge of anomaly detection in IoT–Edge–Cloud networks while reducing	– Collects sensory data at the edge and applies the marching squares algorithm to generate isopleths	The primary focus is on energy efficiency and detection accuracy rather than on dedicated security/privacy	Experiments on an air quality hazardous gas dataset show superior boundary accuracy and lower energy

Reference	Focus / Challenges	Key Techniques & Methods	Security / Privacy Features	Evaluation / Results
	unnecessary data transmission and energy use.	– Employs a filtering mechanism at the edge to transmit only anomaly-relevant data – Uses Kriging spatial interpolation for boundary refinement with mobile sensing nodes	methods.	consumption compared to state-of-the-art techniques.
[33]	Deals with collaborative AI for healthcare applications in IoT–Edge–Main Cloud systems; addresses network congestion, latency in model training, and an expanded attack surface.	– Proposes a collaborative, hierarchically merged training technique – Synthesizes a main cloud model from trained edge model layers – Focuses on complexity reduction without major accuracy compromise	While security challenges (expanded attack surface) are acknowledged, the primary focus is on improving training efficiency and accuracy rather than on in-depth security protocol development.	Demonstrates a significant reduction in main cloud model training times while maintaining high detection accuracy, also addressing some limitations of federated learning approaches.
Proposal	Deals with application extend from IoT to edge like fog servers to cloud servers.	-- Proposes combinations between encryption and integration algorithms to build authenticated encryption algorithms -- Proposed authenticated encryption algorithms compatible with edges networks	The objective is to -- develop Iot/edge AE algorithm ChaCha20-Photon differ from Edge/ cloud AE algorithm Blowfish-SHA-512 -- competitive to the standard AE algorithms	-- ChaCha20-Photon Provides moderate throughput, slightly higher latency, moderate energy consumption and high security, as comparison with strong ChaCha20-Poly1305 --Blowfish-SHA-512 provide moderate throughput, higher latency, higher energy consumption, and high security, as comparison with strong AES-GCM

4. Conclusion

From tables (1) and (4) we can conclude the following: ChaCha20-PHOTON offers efficient, secure, and resource-efficient authenticated encryption when used with constrained resources like IoT devices, embedded systems, and mobile apps. The hybrid ChaCha20-PHOTON algorithm provides (50 Mbps) throughput, (15 Mbps) MAC generation, (2 ms) latency, (100 μJ) energy consumption, and high security, as compared with strong ChaCha20-Poly1305. It balances performance and security by introducing strong encryption and authentication in lightweight settings. Both algorithms ChaCha20-Poly1305 and ChaCha20-PHOTON are highly secure. From tables (2) and (3) we can conclude the following: Blowfish-SHA-512 is a reasonably efficient and secure AE algorithm; this hybrid approach uses Blowfish encryption and SHA-512 authentication to present robust security for applications that prioritize security over performance, it provides (20mbps) throughput, (19mbps) *MAC Generation*, (5 ms) latency, *(*200 μJ) energy consumption, and high security in comparison with strong (AES-GCM).

Future works are how to make the two proposals ChaCha20-PHOTON and Blowfish-SHA-512 much more lightweight by modifying the basic algorithms (ChaCha20, PHOTON,

Blowfish, and SHA-512) and more making it secure by demonstrating the key generation and increasing their complexity.

References

[1] J. CAO, Q. ZHANG and W. SHI, "Edge computing: a primer.," Germany: Springer International Publishing, 2018. DOI:10.1007/978-3-030-02083-5

[2] K. DOLUI and S. K. DATTA, "Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing. ," in *2017 Global Internet of Things Summit (GIoTS)*, Geneva, Switzerland, 2017. DOI:10.1109/GIOTS.2017.8016213

[3] R. S. T. S. V. C. ,. A. B. a. B. S. Shivani Singh, "Machine Learning Assisted Security and Privacy Provisioning for Edge Computing: A Survey," *IEEE Internet of Things Journal,* p. 99, July 2021. DOI:10.1109/jiot.2021.3098051

[4] J. K. A. B. a. S. B. Chahal, "DDoS attacks & defense mechanisms in SDN-enabled cloud: Taxonomy, review and research challenges," *Computer Science Review,* vol. 53, p. 100644, 2024. DOI:10.1016/j.cosrev.2024.100644

[5] C.-H. L. a. H.-H. S. a. L.-C. Wang, "Eavesdropping prevention for heterogeneous Internet of Things systems," in *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, Las Vegas, NV, USA, 2018. DOI:10.1109/CCNC.2018.8319297

[6] D. D. R. P. a. N. D. S. Parikh, "Security and privacy issues in cloud, fog and edge computing," *Procedia Computer Science,* vol. 160, no. 1, p. 734–739, 2019. DOI:10.1016/j.procs.2019.11.018

[7] Y. L. X. H. X. D. L. Xiao, "Cloud-based malware detection game for mobile devices with offloading," *IEEE Transactions on Mobile Computing,* vol. 16, no. 10, p. 2742–2750, 2017. DOI:10.1109/TMC.2017.2687918

[8] L. X. a. H. V. P. G. Han, "Two-dimensional anti-jamming communication based on deep reinforcement learning," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, LA, USA, 2017. DOI:10.1109/ICASSP.2017.7952524

[9] M. M. A. I. A. a. E. B. Alani, "ARP-PROBE: An ARP spoofing detector for Internet of Things networks using explainable deep learning," *Internet of Things,* vol. 23 , p. 100861, 2023. DOI:10.1016/j.iot.2023.100861

[10] F. e. a. ALHARBI, "Collaborative client-side dns cache poisoning attack," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, 2019. DOI:10.1109/INFOCOM.2019.8737514

[11] Z. T. a. M. Khemakhem, "Sybil nodes as a mitigation strategy against sybil attack," *Procedia Computer Science,* vol. 32, no. 1, p. 135– 1140, 2014. DOI:10.1016/j.procs.2014.05.544

[12] P. e. a. . SHANMUGARAJA, "An Efficient Clustered M-path Sinkhole Attack Detection (MSAD) Algorithm for Wireless Sensor Networks.," *Adhoc & Sensor Wireless Networks ,* vol. 55, pp. 23-43, 2023. DOI: 10.32908/ahswn.v55.8849

[13] S. R. a. T. V. L. Wallgren, "Routing attacks and countermeasures in the rpl-based internet of things," *International Journal of Distributed Sensor Networks,* vol. 9, no. 8, p. 794326, 2013. DOI:10.1155/2013/794326

[14] S. T. a. K. D. M. Prasad, "Wormhole attack detection in ad hoc network using machine learning technique," in *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, Kanpur, India, 2019. DOI:10.1109/icccnt45670.2019.8944634

[15] R. GOENKA, M. CHAWLA and N. TIWARI, " A comprehensive survey of phishing: Mediums, intended targets, attack and defence techniques and a novel taxonomy.," *International Journal of Information Security,* vol. 23, no. 2, pp. 819-848, 2024. DOI:10.1007/s10207-023-00768-x

[16] E. B. H. M. a. T. Y. Kavun, "A Survey on Authenticated Encryption--ASIC Designer's Perspective," *ACM Computing Surveys (CSUR),* vol. 50, no. 6, pp. 1-21, 2017. DOI:10.1145/3131276

[17] J. a. Y. L. Katz, Introduction to Modern Cryptography., 3rd Edition ed., New York: Taylor &

Francis Group Logo, 2020, p. 648.

[18] L. S. . JAMIL, "Developing Blockchain Algorithms in the IoT Network to Secure DataIntegrity and System Scalability," *Iraqi Journal of Science,* vol. 65, no. 6, pp. 3403-3418, 2024. DOI:10.24996/ijs.2024.65.6.35

[19] M. D. e. a. . ABDUL-JABBAR, " A Key Based Hybrid Approach for Privacy and Integrity in Multi-Cloud," *Iraqi Journal of Science,* vol. 64, no. 11, pp. 5952-5963., 2023. DOI:10.24996/ijs.2023.64.11.39

[20] S. N. HUSSEIN, A. H. OBAID and A. . JABBAR, "Encryption Symmetric secret Key in Wireless Sensor Network Using AES Algorithm," *Iraqi Journal of Science,* vol. 63, no. 11, pp. 5037-5045., 2022. DOI:10.24996/ijs.2022.63.11.38

[21] C.-H. L. a. H.-H. S. a. L.-C. Wang, " Authenticated encryption schemes: A systematic review," *IEEE Access,* vol. 10, no. 1, pp. 14739-14766., 2022. DOI:10.1109/access.2022.3147201

[22] N. Febitri, H. Witriyono, M. Muntahanah, and M. Marhalim, "Application of AES 256 Cryptography Algorithm OCB Mode on Student Data", *j.komputer, j.informasi, j.teknologi*, vol. 3, no. 2, pp. 423–432, Dec. 2023. DOI: 10.53697/jkomitek.v3i2.1478

[23] J.-P. P. J. a. S. N. Aumasson, "NORX: Parallel and Scalable AEAD," in *European Symposium on Research in Computer Security (2014)*, Cham, 2014. DOI:10.1007/978-3-319-11212-1_2

[24] S. A. L. a. Y. L. Gueron, "AES-GCM-SIV: Nonce Misuse-Resistant Authenticated Encryption," *RFC 8452,* pp. 1-42, 2019. DOI:10.17487/RFC8452

[25] F. D. A. S. a. G. S. Santis, "ChaCha20-Poly1305 authenticated encryption for high-speed embedded IoT applications," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, Lausanne, Switzerland, 15 May 2017. DOI:10.23919/DATE.2017.7927078

[26] C. E. M. M. F. &. S. M. . Dobraunig, "Ascon v1. 2: Lightweight authenticated encryption and hashing," *Journal of Cryptology,* vol. 34, no. 1, pp. 1-42., 2021. DOI:10.1007/s00145-021-09398-9

[27] K. P. M. M. M. a. G. M. Koleci, "A Flexible NTT-Based Multiplier for Post-Quantum Cryptography," *IEEE Acces,* vol. 11, pp. 3338-3351, 2023. DOI:10.1109/ACCESS.2023.3234816

[28] Heidari A, Navimipour NJ, Jamali MA, Akbarpour S. A green, secure, and deep intelligent method for dynamic IoT-edge-cloud offloading scenarios. Sustainable Computing: Informatics and Systems. 2023 Apr 1;38:100859.

[29] Arzovs A, Judvaitis J, Nesenbergs K, Selavo L. Distributed learning in the iot–edge–cloud continuum. Machine Learning and Knowledge Extraction. 2024 Feb 1;6(1):283-315.

[30] Seifelnasr M, AlTawy R, Youssef A, Ghadafi E. Privacy-Preserving Mutual Authentication Protocol With Forward Secrecy for IoT–Edge–Cloud. IEEE Internet of Things Journal. 2023 Sep 26;11(5):8105-17.

[31] Chaudhary A, Peddoju SK, Chouhan V. Secure authentication and reliable cloud storage scheme for IoT-edge-cloud integration. Journal of Grid Computing. 2023 Sep;21(3):35.

[32] Li Y, Zhou Z, Xue X, Zhao D, Hung PC. Accurate anomaly detection with energy efficiency in IoT–Edge–Cloud collaborative networks. IEEE Internet of Things Journal. 2023 May 8;10(19):16959-74.

[33] Gupta L. Collaborative Edge-Cloud AI for IoT Driven Secure Healthcare System. In2023 IEEE International Systems Conference (SysCon) 2023 Apr 17 (pp. 1-8). IEEE.

[34] V. R. Kebande, "Extended-Chacha20 Stream Cipher With Enhanced Quarter Round Function," in *IEEE Access*, vol. 11, pp. 114220-114237, 2023, doi: 10.1109/ACCESS.2023.3324612.

[35] A. J. Hendrickson and D. P. Haefner, "Photon Counting Histogram Expectation Maximization Algorithm for Characterization of Deep Sub-Electron Read Noise Sensors," in *IEEE Journal of the Electron Devices Society*, vol. 11, pp. 367-375, 2023, doi: 10.1109/JEDS.2023.3290106.

[36] R. Anusha Padmavathi, K. S. Dhanalakshmi and K. Kalai Selvi, "VLSI Secure communication:

Advancements in Blowfish Algorithm for Secure Data Transmission," *2024 7th International Conference on Circuit Power and Computing Technologies (ICCPCT)*, Kollam, India, 2024, pp. 1858-1863, doi: 10.1109/ICCPCT61902.2024.10673112.

[37] A. Gupta, N. Banakar, C. K. N, A. M and P. U, "Design and Implementation of an Efficient Fingerprint Authentication Algorithm using SHA-512," *2024 Third International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE)*, Ballari, India, 2024, pp. 1-7, doi: 10.1109/ICDCECE60827.2024.10548466.